Shading by Spherical Linear Interpolation using De Moivre’s Formula

Anders Hast
Creative Media Lab
University of Gävle, Sweden
aht@hig.se

Tony Barrera
Cycore AB, Sweden
tony.barrera@spray.se

Ewert Bengtsson
Centre For Image Analysis
Uppsala University, Sweden
ewert@cb.uu.se

ABSTRACT

In the classical shading algorithm according to Phong, the normal is interpolated across the scanline, requiring a computationally expensive normalization in the inner loop. In the simplified and faster method by Gouraud, the intensity is interpolated instead, leading to faster but less accurate shading. In this paper we use a third way of doing the interpolation, namely spherical linear interpolation of the normals across the scanline. This has been explored before, however, the shading computation requires the evaluation of a cosine in the inner loop and this is too expensive to be efficient. By reformulating the original approach in a suitable way, De Moivre’s formula can be used directly for computing the intensity so that no normalization is needed. Hence, no trigonometric functions, divisions or square roots are necessary to compute in the inner loop. Unfortunately the setup for each scanline will be rather slow unless some efficient reformulation of the necessary trigonometric calculations can be found. We suggest this problem for future research.

Keywords
Shading, Normalization, Slerp, De Moivres formula.

1. INTRODUCTION

Shading is a graphical technique for rendering more realistic images of 3D objects. Two widely used techniques are known as Gouraud [Gou71] and Phong Shading [Pho75]. In Gouraud shading, the intensities at the vertices of the polygon are calculated first. Then bilinear interpolation is used in order to obtain intermediate intensities at the interior of the polygon. In Phong shading, bi-linear interpolation of the normal vector is performed over the polygon in order to obtain intermediate normals on the interior of the polygon. These interpolated normals are then used in the lighting calculation both for the diffuse and specular light. Gouraud shading is much faster than Phong shading but suffers from the Mach band effect and handles specular reflections poorly. Phong shading will produce more accurate highlights than Gouraud shading. However, vector interpolation is computationally expensive, since the interpolated normal must be normalized. Otherwise, the result will not be much different from Gouraud shading. The normalization process includes both a division and a square root and both are operations we would like to avoid in the inner loop.

In this paper we will propose the use of a third approach, where the normals are interpolated linearly spherically over the scanline, i.e. with equal angle increments rather than equal linear increments. This will give shading of the same quality as Phong shading. Formulating this in an appropriate way, the normalization of interpolated normals can be avoided over each scanline. Hence, no division and square roots are necessary in the inner loop. However, trigonometric functions need to be computed in the inner loop. This paper will show that by reformulating the shading interpolation using De Moivre’s formula, these trigonometric functions can be eliminated and completely avoided in the inner loop. However, the normals on the edges must be

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WSCG SHORT PAPERS proceedings
WSCG’2003, February 3-7, 2003, Plzen, Czech Republic.
Copyright UNION Agency – Science Press

normalized. The resulting algorithm is fast in the inner loop for the per pixel computation. Nonetheless, it will still require extensive computations for the setup of each scanline and some solutions are proposed for future work in this paper.

2. PREVIOUS WORK

Kuijk and Blake [Kuij89] showed how angular interpolation could be used for faster Phong shading. They use spherical trigonometry to derive an equation for how both the normal and the vector in the direction to the light source varies over the polygon. A cosine has to be evaluated for each pixel. However, they propose a quadratic approximation that will make the evaluation faster. Abbas et al. elaborates this idea further for a suitable hardware implementation. A number of other approaches have been introduced which does not use spherical interpolation. Nevertheless, they should be mentioned, since they are relatively fast. They are based on quadratic approximation of the shading curve and are therefore different from the interpolation approach described in this paper. Bishop and Weimer [Bis86] used a Taylor series expansion of the Phong equation in order to obtain a second order approximation. Seiler [Sei98] and Lee and Jen [Lee01] elaborated an idea by Kirk and Voorhies [Kir90], which is based on the fact that a quadratic shading surface can be determined from six sample points over the polygon, (i.e. the intensity at the vertices and the edge midpoints).

3. REFORMULATION OF THE SHADING COMPUTATION

Shoemake [Sho85] showed how spherical linear interpolation (slerp) could be done, by using quaternions. The slerp-formula could also be used for rotating vectors. The spherical linear interpolation of the intermediate normal between the normals $\vec{N}_a$ and $\vec{N}_b$ is:

$$\vec{N}(t) = \vec{N}_a \frac{\sin((1-t)\theta)}{\sin\theta} + \vec{N}_b \frac{\sin(t\theta)}{\sin\theta}, \tag{1}$$

where $t \in [0,1]$ and θ is the angle between $\vec{N}_a$ and $\vec{N}_b$. This equation can be expanded using trigonometric rules in the following way:

$$\begin{aligned}\vec{N}(t) = \vec{N}_a \frac{\sin\theta\cos(t\theta) - \cos\theta\sin(t\theta)}{\sin\theta} &+ \\ \vec{N}_b \frac{\sin(t\theta)}{\sin\theta} &=\end{aligned} \tag{2}$$

$$\begin{aligned}\vec{N}_a \cos(t\theta) - \vec{N}_a \frac{\cos\theta\sin(t\theta)}{\sin\theta} &+ \\ \vec{N}_b \frac{\sin(t\theta)}{\sin\theta}&.\end{aligned} \tag{3}$$

Note, that $\cos\theta = \vec{N}_a \bullet \vec{N}_b$ and the trigonometric unity gives that $\sin\theta = \sqrt{1-\cos^2\theta}$. Hence, equation (3) can be rewritten as:

$$\begin{aligned}\vec{N}(t) = \vec{N}_a \cos(t\theta) &+ \\ \frac{\vec{N}_b - \vec{N}_a(\vec{N}_a \bullet \vec{N}_b)}{\sqrt{1-(\vec{N}_a \bullet \vec{N}_b)^2}} &\sin(t\theta).\end{aligned} \tag{4}$$

Finally we will show that the last term of equation (4) is actually the tangent vector received by applying the first step in the Gram-Schmidt orthogonalization algorithm [Nic95]:

$$\vec{N}_t = \vec{N}_b - \frac{(\vec{N}_a \bullet \vec{N}_b)}{\left\|\vec{N}_a\right\|^2}\vec{N}_a, \tag{5}$$

where $\left\|\vec{N}_a\right\|^2 = 1$, since it is required that the normals at the edges are normalized. The tangent vector $\vec{N}_t$ is by definition orthogonal to $\vec{N}_a$. Thus we have:

$$\vec{N}_t = \vec{N}_b - (\vec{N}_a \bullet \vec{N}_b)\vec{N}_a. \tag{6}$$

This is exactly what we have in the numerator of the last term in equation (4). We will now show that the denominator of the last term in equation (4) is actually the norm of the numerator. This implies that the term before sine in equation (4) must be the normalized tangent vector. The norm is:

$$\left\|\vec{N}_t\right\| = \sqrt{(\vec{N}_b - (\vec{N}_a \bullet \vec{N}_b)\vec{N}_a)^2} \tag{7}$$

$$= \sqrt{1 - 2(\vec{N}_a \bullet \vec{N}_b)^2 + (\vec{N}_a \bullet \vec{N}_b)^2} \tag{8}$$

$$= \sqrt{1-(\vec{N}_a \bullet \vec{N}_b)^2}\ . \tag{9}$$

Hence, we can write equation (1) as:

$$\vec{N}(t) = \vec{N}_a \cos(t\theta) + \vec{N}_t \sin(t\theta). \tag{10}$$

Shading by Angular Interpolation

One of the mentioned drawbacks with Phong shading is that it is necessary to normalize the normal at each pixel. Another effect of linear interpolation is that the angle between each normal on the scan line will not be the same. By using slerp both these problems are solved. However, it is necessary to evaluate both a sine and cosine in the inner loop using slerp. This is not necessary if the De Moivre's formula is used, as we shall show in the next section. Nonetheless, we must first state what is necessary for using slerp on a scanline.

First we need the angle between $\vec{N}_a$ and $\vec{N}_b$, which is denoted:

$$\theta = \cos^{-1}(\vec{N}_a \bullet \vec{N}_b). \qquad (11)$$

If a scan line has k pixels then the angle between each new interpolated normal denoted K_θ is:

$$K_\theta = \frac{\theta}{k}. \qquad (12)$$

Subsequently we rewrite equation (10) into a form which suites our incremental scanline shading scheme:

$$\vec{N}(n) = \vec{N}_a \cos(nK_\theta) + \vec{N}_t \sin(nK_\theta), \qquad (13)$$

where n is the n'th pixel along the scanline currently being shaded. As noted by Duff [Duf79], it is possible to make the computation more efficiently by computing the dot product $\vec{N} \bullet \vec{L}$ directly instead of interpolating the normal and then computing the dot product.

This is also possible for spherically interpolated shading. For the diffuse intensity the computation is:

$$I_d(n) = I_a \cos(nK_\theta) + I_t \sin(nK_\theta), \qquad (14)$$

where

$$I_a = \vec{L} \bullet \vec{N}_a, \qquad (15)$$

$$I_t = \vec{L} \bullet \vec{N}_t. \qquad (16)$$

This will make the computation noticeably faster since we will compute scalars instead of vectors. However, it will still be rather slow since the sine and cosine functions are even more computationally expensive than the original division and square root in ordinary normalization. Obviously, this approach will only be useful if we can derive a faster way of evaluating equation (14) along a scanline.

The De Moivre's formula

Complex numbers are defined in an orthogonal system where the base vectors are [1,0] and [0,i]. Hence, they could be regarded as 2D vectors in euclidian space, if we let:

$$v = [\Re(Z), \Im(Z)]. \qquad (17)$$

The De Moivre's formula states that:

$$(\cos\phi + i\sin\phi)^n = \cos(n\phi) + i\sin(n\phi). \qquad (18)$$

Note that the right part of equation (18) is similar to equation (13) in the way that both have a cosine and sine term, which are multiplied with vectors that are orthogonal to each other.

Let Z be a complex number computed by:

$$Z = \cos(n\phi) + i\sin(n\phi). \qquad (19)$$

Furthermore, we treat complex numbers as if they were vectors in 2D-space. The diffuse intensity is computed by the dot product between such a vector and an ordinary vector, which in this case is the intensities. The dot product is:

$$\begin{aligned} I_d &= [I_a, I_t] \bullet [\Re(Z), \Im(Z)] = \\ &\quad I_a\Re(Z) + I_t\Im(Z). \end{aligned} \qquad (20)$$

Expanding the right part gives:

$$I_d(n) = I_a \cos(n\phi) + I_t \sin(n\phi). \qquad (21)$$

This is exactly the same as equation (11) if we substitute ϕ with K_θ. Thus, we can compute the diffuse intensity as:

$$I_d(n) = [I_a, I_t] \bullet Z^n. \qquad (22)$$

The total cost for computing the diffuse intensity is one complex multiplication and one dot product. Moreover, the complex multiplication or square is computed by:

$$Z^2 = (x, y)^2 = (x^2 - y^2, 2xy). \qquad (23)$$

Thus, the total cost is six multiplications and one addition and one subtraction for each pixel.

4. DISCUSSION

The approach that is proposed in this paper is not an approximating scheme like the quadratic shading approaches mentioned in the previous work section. It is truly an interpolating scheme, which interpolates the normal via angular interpolation and then the intensity is computed. The quadratic schemes will be very fast in the inner loop but a substantial setup is required for each polygon. The setup for the proposed approach will be quite small for each polygon. However, there is a substantial setup before each scanline. Moreover, it will be very fast for the inner loop since there are no divisions and square roots involved. If a hardware implementation of the proposed scheme is made, then the total computational cost could be reduced, by implementing a complex multiplication and a dot product directly in hardware. Unfortunately, most of the speedup gained, will be lost by having to use the inverse cosine, sine and cosine functions in the setup for each scanline. This problem should be solved.

One solution would be to use table lookups. However, we are currently investigating the use of Maclaurin polynomials for approximation of the trigonometric functions needed.

It is also important to solve the problem when θ is small, since $\vec{N}_t$ is not defined for θ=0. A simple solution would be to let Z=(1,0) for θ<ε, where ε is a threshold value.

The specular light can be computed in a similar way. The Z computed for the diffuse light can of course be reused for the specular light.

It should also be noted that De Moivre's formula could be used to interpolate normals directly:

$$N(n) = [N_a, N_t] \bullet Z^n . \tag{24}$$

However, this approach could not compete with doing rotation using a rotation matrix.

Modern graphics processors come with programmable pixel shaders. In order to be able to use the flexibility provided by these, it is required that a normal is available for each pixel. If fast evaluation of trigonometric functions could be done, by using Maclaurin polynomials, then it would even be feasible to use a rotation matrix for normal interpolation in hardware.

The shading curve produced by spherically interpolated shading will be almost identical to the curve obtained by using Phong shading. In figure 1, the difference in intensity between Phong shading and spherical linear interpolated shading is plotted. The curve obtained is sinus like. Note the scale on the y-axis, the difference is quite small and will in practice not be visible. It is therefore no point in comparing shaded images. Nonetheless, a shaded torus using slerp for both the diffuse and specular light is shown in figure 2.

It should also be pointed out that the interpolation scheme presented is using a distant light source. How point light sources could be implemented using the proposed algorithm should be ascertained in future research.

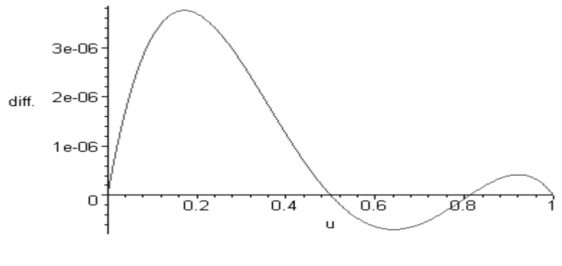

Figure 1. Difference in intensity between Phong shading and shading using slerp.

5. CONCLUSIONS

By using spherical linear interpolation of the normal, or rather the shading curve together with De Moivre's formula, it is possible to get a shading interpolation along the scanline, which does not include any square roots or divisions. However, in order to make this approach useful, there are still some computations that must be made more efficient. We propose this as a research problem in this paper, and we believe that Maclaurin polynomials could be used for this purpose.

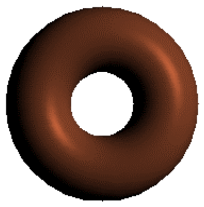

Figure 2. A Torus shaded using slerp.

6. REFERENCES

[Abb00]A. M. Abbas, L. Szirmay-Kalos, T. Horvath, Hardware Implementation of Phong Shading using Spherical Interpolation, Periodica Polytechnica, Vol. 44, Nos 3-4, 2000.

[Bis86] G. Bishop, D. M. Weimer, Fast Phong Shading, Computer Graphics, vol. 20, No 4, pp. 103-106, 1986.

[Duf79] T. Duff, Smoothly Shaded Renderings of Polyhedral Objects on Raster Displays, ACM, Computer Graphics, Vol. 13, pp. 270-275, 1979.

[Gou71] H. Gouraud, Continuous Shading of Curved Surfaces, IEEE transactions on computers vol. c-20, No 6, June 1971.

[Kir90] D. Kirk, D. Voorhies, The Rendering Architecture of the DN10000VS, Computer Graphics vol. 24, pp. 299-307, August 1990.

[Kui89] A. A. M. Kuijk, E. H. Blake, Faster Phong Shading via Angular Interpolation, Computer Graphics Forum, vol. 8, No 4, pp. 315-324 1989.

[Lee01] Y. C. Lee, C. W. Jen, Improved Quadratic Normal Vector Interpolation for Realistic shading, The Visual Computer, 17, pp. 337-352, 2001.

[Nic95] W. K. Nicholson, Linear Algebra With Applications, PWS Publishing Company, Third Edition, pp. 275, 1995.

[Pho75] B. T. Phong, Illumination for Computer Generated Pictures, Communications of the ACM, Vol. 18, No 6, June 1975.

[Sei98] L. Seiler, Quadratic Interpolation for Near-Phong Quality Shading, SIGGRAPH 98: abstracts and applications, pp. 268, 1998.

[Sho85] Ken Shoemake, Animating rotation with quaternion curves, ACM SIGGRAPH,Volume 19 Issue 3, pp. 245-254, July 1985.